



LINUX Expert

Hackers continue to find new ways to penetrate your defences and compromise your operating system.

Jon Thompson explains how to spot and seal security weaknesses in Linux and keep hackers at bay



NOVELL STICKS WITH LINUX – FOR NOW

After Novell shed 10 per cent of its worldwide staff, stories began circulating that the company was going to drop Linux. The source of this story was the Linux Today website.

"Contrary to what was expected from recent Novell announcements," the story went, "Novell executives are apparently slicing deeply into the Linux heart of the company." But Novell disagrees. Novell's Linux director of marketing Greg Mancusi-Ungaro said, "Novell has a Linux desktop leadership position with Novell Linux Desktop and SuSE 10, and we intend to keep the pedal down hard".

Novell is following Red Hat's lead by building its commercial Linux offering on the new open-source, community-developed distribution OpenSuSE. Red Hat is basing its Enterprise Linux on the free Fedora release.

OpenSuSE has spawned a number of other unofficial OpenSuSE-based distributions. These include the SuSE Performance Enhanced Release (SUPER) project, which aims to speed up the operating system using experimental kernel patches, packages and configurations. So far it has halved boot time, but many users will be more interested in its enhanced features.

Novell, like other commercial distributors, leaves the user to install DVD codecs, which we showed you how to do in last month's *Linux Expert*. SUPER, being a free distribution, has no such restrictions and comes with the codecs preinstalled. The project plans to add the BitTorrent client Azureus, which Novell cannot do because of licensing restrictions.

Giving a hint at the nature of its relationship with its developer community, Novell has already adopted a speed improvement from the SUPER distribution. SUPER preloads OpenOffice.org and Firefox at system startup, so these programs are available quickly when needed. Novell has adopted this feature for OpenSuSE Beta 4, which will preload the GIMP image-editing package, the Mozilla web browser and other KDE applications such as the Konqueror browser and the Kicker application toolbar.

Hackers are a problem, even to Linux experts. Because of the sheer complexity of operating systems such as Linux, there are times when even the best-protected systems are left unintentionally wide open to attack. Despite the best efforts of system administrators and law enforcement agencies, people who break into computers or spread malicious code still pose a threat. With a little forethought, however, you can ensure your computer is an unattractive prospect, even if an attacker can circumvent your firewall, intrusion detection system (IDS) and virus checker.

A scan of a few security mailing lists shows that even old, well-analysed programs can have hidden vulnerabilities lurking within their code. These are usually caused by assumptions made at the start of their development. Security researchers employed by software houses, or working in independent groups, find most of these vulnerabilities and report them to the developers, who produce updates to fix the security holes. There are also private individuals who chase exploitable bugs as a hobby, in much the same way that amateurs might spend their free time tracking down comets and asteroids.

For some, however, the feeling of pride at discovering and publicising a new threat is outweighed by the illicit thrill of knowing they hold the key to untold numbers of systems. These people pose a serious threat, and this month we explain how their techniques can punch through your defences, and how to find out when they have.

BIG ZERO

Let's suppose you've been hacked. You've taken all the right steps, including running a firewall, anti-virus software and an IDS, but it still happened. Programs you know nothing about are running, the network is slow, disk usage is through the roof and the processors face meltdown. You might have forgotten to apply a recently released security patch, but it's possible you've been hit by an exploit kept secret and used only by the person who discovered it. If someone has secret knowledge of a vulnerability, and the ability to break in using it, they have what is known as a 'zero day' exploit.

However they got there, you have an unwanted visitor. The first thing to do is unplug the site router's network connection until you know for sure how many computers have been compromised, and also unplug the suspect systems from the internal network. Linux, unlike Microsoft Windows, is a multi-user operating system. If you can gain access, you can do far more than just connect to shared resources. This makes Linux and other UNIX-like operating systems prime targets, not for viruses, but for being 'rooted', where hackers try to gain superuser access. If they gain that, they can do anything on the system, from launching an

attack against their real victim from your computers, to destroying all evidence that they were ever there.

From the pragmatic system administrator's point of view, an ounce of prevention is worth a pound of cure. You need to make your computers as hacker-unfriendly as possible in the first place. Hackers who can gain root access to a protected box are also capable of realising when the operating system has been conspicuously configured against them. They will be more inclined to attack softer targets, which are less likely to ring alarm bells and track them down. If you make the effort, and they value their liberty, they will attack less well-informed targets.

WHO GOES THERE?

We can divide hackers into three distinct groups based on ability. First and most prevalent are the so-called 'script kiddies', who use potted scripts that exploit known and usually very old vulnerabilities to gain access to as-yet unpatched computers. They deface websites and cause general mayhem. Script kiddies don't tend to have the skills to cover their tracks; they don't generally understand enough to know that everything they do is traceable without special efforts being taken, and they don't appreciate that prosecution may just be a matter of time.

The second group is driven by the desire to find free, anonymous storage space. The reasons for this might be anything from the distribution of pirated software and music to illegal images. An FTP server with anonymous access left open through a firewall is just the ticket. But equally, systems with unpatched vulnerabilities and enough free space inside might become hosts for things the owners would rather not be associated with, however unwittingly.

The third group are professional hackers, and these are employed to break into specific computers for the purposes of industrial espionage and to write the code for botnets. These are networks of unpatched computers, taken over and directed to generate spam or launch distributed denial-of-service (DoS) attacks.

THE BUFFER SLAYER

Regardless of whether a hacker is on the other side of the world or just down the corridor, one of the fastest ways to gain unauthorised access to a networked computer is via a buffer overflow attack. The most remarkable thing about attacks of this nature is that, while a great number of them have already come to light and been patched, they shouldn't be possible at all.

A buffer overflow happens when too much data is packed into a section of memory that is reserved to act as a local storage space. The surplus



data overflows and overwrites adjacent memory locations. This can disrupt other programs, but in skilled hands it can do much worse. Consider the following small ANSI C program:

```
#include <stdio.h>
#include <string.h>

#define BUF_LEN 5

int main(int argc, char **argv){
    char buf[BUF_LEN];
    printf("buffer length: %d\n", BUF_LEN,
        strlen(argv[1]));
    strcpy(buf, argv[1]);
    return(0);
}
```

What this program does is simple, but how it's done could be dangerous in the wrong hands. It accepts a number of arguments from the command line and puts them into the standard array called `argv`. Then it sets up a small buffer called `buf`, which is capable of holding a number of characters. This number is defined by the `BUF_LEN` variable. After printing out a couple of statistics about the length of the buffer and the first argument passed on the command line, it attempts to place this argument into the buffer using the string copy (`strcpy`) statement. It then exits, returning control to the command line.

So far so good, but it's what this program doesn't do that's the interesting thing here. There's an implicit assumption that the user knows not to pass a first argument from the command line that is more than `BUF_LEN` characters in length. So what happens if the argument is longer than it should be? Because it is written in C, the program will execute the `strcpy` statement regardless. If the result of that copy overruns the end of the space in memory reserved for the buffer, then so be it.

C gains its legendary speed in part from a lax attitude when it comes to checking data manipulations, and even allows a few that don't make sense. For example, the onus is placed on the programmer to check that a program doesn't try to place a string into an integer variable. If you deliberately try to place too many characters into the buffer, the program will happily oblige. Enter and save the program under the filename `overflow.c`. Compile and run it as follows:

```
# gcc overflow.c
# ./a.out abcdefg
buffer length: 6 argument length: 7
```

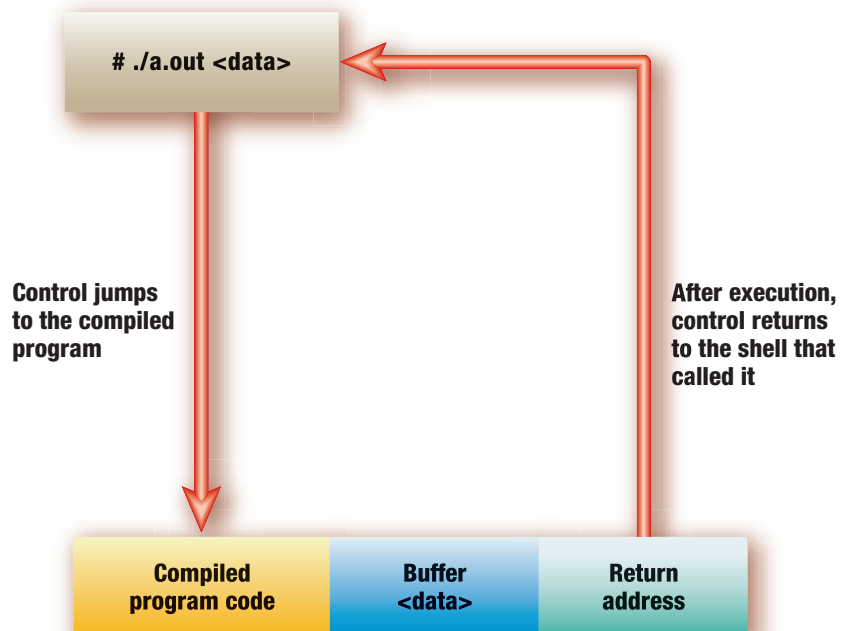
In Linux, like all UNIX-like operating systems, the C compiler will store its output in a file called `a.out` unless otherwise directed. Try entering evermore characters as the first argument to `a.out`. At a certain length of argument (about 28 characters), the program will crash and produce a fault.

You may be wondering why this example of sloppy programming would be of interest to a hacker. Surely all someone can do by passing too much data to a buffer is corrupt the memory reserved for the program as it runs, or perhaps make the system a little unstable? It turns out he can do far more.

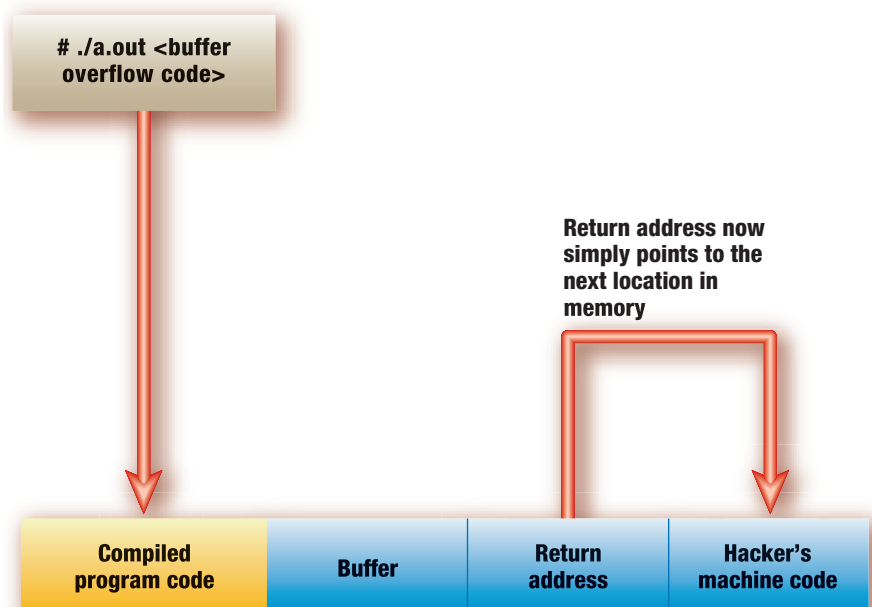
The area of memory beyond the end of the buffer is used to store other information, such as

BUFFER HACKING IN ACTION

Some of the most devastating attacks against online computers work because they have not been patched against exploitable buffer overflow conditions. The short program we mention on the left, when compiled and executed from the command line, accepts an argument and occupies a few bytes of memory as follows:



If an attacker overflows the buffer space (shown in blue) by passing an argument that is too long, he will overwrite the return address and beyond. By doing this carefully, he can make the program execute the rest of the argument when it jumps to the overwritten return address:



The rest of the argument would need to be equally carefully constructed. It must contain machine code. The potential of what this code can do is virtually unlimited, particularly if the program being attacked happens to be a service on a remote computer that runs as root. The resulting code might call a shell and pop up a command line with root access.



the return address for when the program exits. It's this memory location that is of supreme interest to anyone performing a buffer overflow attack. The reason Linux gave a segmentation fault when we entered a certain length of argument is because it took that many bytes to overwrite the return address. In this case, it was overwritten with an illegal value. Overwrite this with a legal value and you can, when the program attempts to exit, force it to jump to anywhere you like. It can be made to return not to the command line but to a piece of handcrafted code elsewhere.

One option is to have it simply jump to the very next memory location, which contains the rest of the overflowed data. Writers of buffer overflow attacks carefully craft this to contain specific machine code instructions, which the exiting program is forced to obey. This code might have the hijacked program call a shell, presenting the hacker with a command line owned by the account that started the subverted program. It might easily be far more destructive.

If the program being overflowed is a service on a remote computer and happens to be owned by root, the shell will have superuser privileges. This might happen without alerting the firewall – assuming it's a computer that can be seen from the internet – and without the IDS taking an interest, if the IDS hasn't been updated recently or if the attack is a zero day exploit. The payload injected in this way is usually called 'shellcode' because of its tendency to open a shell. Clearly, this is something you don't want happening to your systems, as it makes a mockery of all your security efforts.

Incidentally, C contains a better statement that we could have used in place of strcpy, called strncpy. It takes a third parameter, giving the maximum number of characters it should copy into the buffer. By making this value BUF_LEN, the overflow can't happen, regardless of the value that BUF_LEN subsequently takes.

CRAZY LIKE A BOX

So what are the signs that someone has broken into your Linux box? Some degree of scoping you out as a target will have taken place before the attack, so there's a good chance your IDS will have alerted you to suspicious port scanning or connection activity. But you should stay on alert after you think your would-be intruder has given up because of this, just in case.

It's not unusual to receive email from other administrators warning you of general threats.

On occasion, you might get an angry note from someone claiming your systems are spamming or launching DoS attacks. They might even accuse you of hacking. In any case, the response should be the same: start an investigation. You could begin by asking to see their log files for evidence of a security breach.

If you've been infiltrated, there's a good chance your visitor will have been sniffing for passwords that will enable him to come and go inconspicuously, posing as a variety of other users. To do

this, he'll need to put the network interface card on the computer he's occupying into promiscuous mode. You can check this by entering the command `ifconfig -a` and looking for the string `PROMISC` in the output. If you're not running a network-based intrusion detection system on the computer in question, which by its very nature needs to put the network card into promiscuous mode, there's a good chance the presence of a promiscuous card means that you have discovered someone sniffing traffic.

Another dead giveaway is a new user in the `/etc/passwd` file. In any organisation, even a small company, it's a good idea to have a procedure whereby the system administrator alone has root access. Only he can add accounts to the system. If he hasn't added the user, there's a problem.

A dramatic decrease in the amount of disk space returned by the `df` command is a good indication that someone is using your computers to store pirated software, so monitor it regularly with the `df` command. Use the `-h` option to view the partition sizes in human-readable values such as M for megabytes and G for gigabytes:

```
# df -h
Filesystem Size Used Avail Use%
Mounted on
/dev/hda1 3.0G 558M 2.3G 20% /
tmpfs 62M 0 62M 0% /dev/shm
/dev/hda7 8.4G 33M 8.0G 1% /home
/dev/hda6 2.0G 33M 1.9G 2% /tmp
/dev/hda5 4.0G 224M 3.6G 6% /var
```

As important as it is to isolate physically a suspected 'dirty' computer from the network, don't be tempted simply to shut it down or reboot it just yet. Take a look at the files stored under directories in `/etc/rc.d`. If there are new scripts here – you can tell by entering `ls -l` and checking the dates for new ones – they might have been left as a nasty surprise for when you shut down or reboot the system. Move any you find to the `/tmp` directory for later scrutiny.

The question of whether or not you should report the attack is, surprisingly, still open to debate. The National Hi-Tech Crime Unit investigates and prosecutes cyber attacks that originate internally and externally. One problem they face is that, by disclosing an attack, many companies think they might damage their reputations. The flipside of this argument is that the company could not have prevented the break-in, because it happened

despite following all the advice given by authorities in internet security and having taken all reasonable steps to ensure its online safety. The blame might actually be down to the person who wrote the vulnerable software, who unwittingly subverted all the company's efforts to stay safe.

AT YOUR COMMANDS

After an attack is confirmed, and you've isolated the system physically, you need to conduct an autopsy and back up any data you need before reformatting the system. It's a good idea to try to understand what's been changed. But there is a chance that the intruder will have replaced innocent commands such as `ls` with those that execute malicious code. More likely, they may contain subtle reprogramming designed to make them hide certain files, perhaps those belonging to an uploaded rootkit, which we'll look at later. Clearly, running an `ls` command as root means the subverted version will run with root's privileges, too. This is something to be avoided.

To get around this problem, make a CD containing the files stored in the `/bin`, `/usr/bin`, `/usr/local/bin`, `/usr/local/sbin` and `/usr/sbin` directories. Keep it in a safe place for emergencies. If you haven't installed any programs in `/usr/local` directories, these might not exist on your system. The `sbin` directories contain the binaries that can be run only by the superuser. If you suspect a system, mount the CD and run commands from it directly. Don't add these directories to your path for convenience; instead, type the entire path to the known good read-only versions. If you just need commands such as `ls`, `grep`, `file`, `find` and so on, these should be small enough to fit on a floppy disk. Ensure that you flip the write-protect tab across to prevent interference or accidentally overwriting the unhacked files on the floppy.

One of the best tools for confirming a suspected attack is `syslog`. This is the first thing a competent hacker must mess with if he's to stand a chance of getting away with abusing your computer. Log cleaning is the process by which an intruder uses uploaded tools to cover his tracks. There are a number of steps one can take to ensure that intruders cannot tamper with `syslog`, and that might make them realise they've left evidence that cannot be cleaned up. Some administrators write `syslog`'s output to a medium that doesn't allow for easy manipulation. If your distribution supports extended file attributes, you can set the `syslog` message file to be append-only, for instance. A wider-reaching solution is to use a file integrity checker, as we showed you in *Linux Expert*, *Shopper* January 2005. AIDE, for example, is bundled with most mainstream Linux distributions.

In hiding his tracks, an intruder might decide to delete his working files, uploaded executables or even `syslog` files. AIDE can tell us after the fact. However, most people don't realise that the Linux kernel doesn't delete the file until the number of programs with a handle on it falls to zero. One particular program might delete a file, but if another has it open too it will remain until this second program exits. If an intruder uploads a rootkit program that opens and deletes a working file then until that rootkit program ends, the file will still be available to us to copy to a safe place.

The `lsf` command shows us which files a program has open and will show which are marked as being deleted:

home What your business really needs to know - top tips Text only site

nhtcu National Hi-Tech Crime Unit A National Crime Squad multi-agency initiative click here

Business & Industry General Public Media Centre FAQs

Welcome to the National Hi-Tech Crime Unit website

Mission To combat national and transnational serious and organised hi-tech crime within, or which impacts upon the United Kingdom

Vision Through sustained leadership and focus on high profile and high impact cases

The UK's National Hi-Tech Crime Unit exists to investigate serious or organised computer crime, but its website (www.nhtcu.org) is also a goldmine of advice for keeping your system safe online

Search

News

NHTCU ALERTS PUBLIC TO FAKE E-MAIL SCAM

The NHTCU is warning the public to avoid falling victim to an ongoing mass virus attack by e-mail wherein computer users receive unsolicited e-mails purportedly sent by the NHTCU. These scam e-mails tell the recipients that their internet use has been monitored and that they have accessed illegal web sites. The e-mails then direct recipients to open an attachment and answer questions. These e-mails did not come from the NHTCU. Anybody who receives such an email should delete it without opening it.


```
# lsof -c hack
prog 1430    root    cwd      DIR    3,4   1024   20197 /dev/
prog 1430    root    mem      REG    3,4   25345  40255 /dev/hack (deleted)
prog 1430    root    mem      REG    3,7  104836 40258 /lib/ld-2.1.3.so
prog 1430    root    mem      REG    3,7   39258 28685 /lib/libc-2.1.3.so
prog 1430    root    lu       REG    3,4    5435  54343 /tmp/output (deleted)
prog 1430    root    5u       REG    3,5   54323  86737 /tmp/gxs3
```

You don't have to give the full path name to the program, as lsof deals directly with the kernel's knowledge of the running system. The kernel, in this example, knows that hack is running so lsof reports on files that have been opened by hack. The suspicious program (/dev/hack) has deleted itself. It certainly is suspicious, too, being hidden away in /dev. There should only be device descriptions in this directory, so any executables there are suspect. Only users with root-level authorisation can write to /dev/, so if you find strange executables in this directory you can be sure someone has attained root-level access on your system.

Even though the executable has deleted itself from the file system, the running image still has a handle on it. Because of this, undeleting the program is just a case of navigating to /proc/1430. The directory is taken from the second column of lsof's output, which is the process number. You can find lots of data about the process in this directory. The program is stored in the exe file. Copy the exe file somewhere else, should you be expert enough to analyse a binary. Libc is running. This indicates that the hack program is a dynamically linked program, which requires the libc library. The ld entry

exists because ld is used to load shared libraries. If hack was a statically linked binary, capable of running independently from programming libraries, you wouldn't see these two entries.

The fourth column gives the directory that contains a cache of the program's inputs and outputs. If the entry is a number followed by a u, it's the name of a file handle. File handles are stored in the fd directory, which stands for file descriptors. The first three possible entries (0, 1 and 2) correspond to the standard file handles. These are stdin for standard input (what you type), stdout for standard output (text returned to the screen) and stderr for error messages.

In this case, the available file handles are called 1u and 5u. They represent the files /tmp/output and /tmp/gxs3. You can read these file handles with the cat command by dropping the trailing u:

```
# cat /proc/1430/fd/5
```

Entry 1 might contain any output that the hack program would normally send to the screen, but the entry called 5 is more interesting because it probably contains data being sent to a file, which

could be passwords or any other information that hack has been programmed to collect. We can deduce this because it has not been deleted.

TRAFFIC JAM

To avoid detection, an attacker can also use an encrypted tunnel. Then any lurking administrators won't be able to sniff suspicious traffic using utilities such as tcpdump or Ethereal. One way to stop the attacker is to configure your firewall strictly, denying all traffic in both directions that tries to use ports other than those you know are completely necessary. Reviewing your firewall rules will help keep out people who use potted exploits that rely on encrypted links to services they've installed, and will make the intruder's job harder.

If an attacker can gain write access to the /etc/inetd.conf file, he can add a service that opens a shell when he connects to it, rather than calling a daemon that provides a service. Connecting via Telnet to the relevant port will give a login prompt or even a command line. To catch this, check the modification time of the inetd.conf file, and use a file integrity checker to alert you if this changes.

Attacks that use modified kernel modules to let an intruder back in will trigger your file integrity checking system, as will a kernel modification. If the kernel is modified, however, perform a rebuild as soon as possible. The intruder may have compiled a code that could do anything the moment you seem to be looking for hacking evidence.

As mentioned earlier, an intruder will sometimes bring with him a set of tools called a rootkit. This is designed to hand him access to the root account, and might contain bogus versions of real commands, utilities for cracking passwords and more besides. As an example, if he's sniffing for passwords to decrypt, he might deploy a version of ifconfig for you to run unwittingly that doesn't display the PROMISC flag, for instance. All the more reason to create a CD of known good commands to use in an emergency.

PATCH WORK

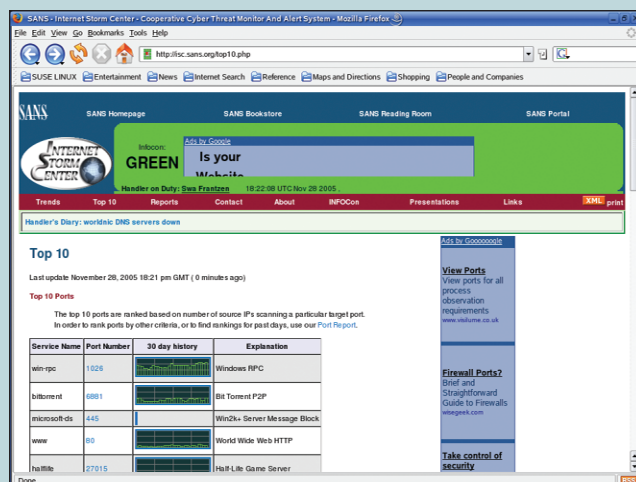
Clearly, there's going to be a constant need to ensure that all your firewall and anti-virus systems are all up to date. It's just as important to keep up to date with the latest security patches to ensure that you have the very best protection from threats that aren't covered by these three lines of defence. Just as Microsoft Windows XP has an automatic updating facility, so do the major Linux distributions.

Unless you work with teams of security specialists, you won't have the time and resources to monitor your computers 24 hours a day. With a little effort, though, you can still ensure that your systems appear a deeply unattractive prospect for even the most ardent hacker. If your site is targeted by a skilled hacker, the best you can expect is to notice the attack and clean up afterwards. Running an IDS can help. We covered this in *Linux Expert*, *Shopper* January 2005, and a PDF of this article is included on this month's cover disc. 

ADVANCED WARNING SITES

Several groups have sprung up over the years to coordinate the fight against hacking. Carnegie Mellon University's Computer Emergency Response Team (CERT, www.cert.org) was formed in 1988. From a control centre in the university's Software Engineering Institute, CERT provides a rallying point for announcements of all the latest vulnerabilities and tracks hacking activities and trends. It also publishes advice on staying safe from hackers and how to write safe code, as well as working with official bodies to improve security standards.

A more informal announcement system is provided by www.securityfocus.com via its Bugtraq mailing lists. These cover everything from announcements of major incidents to the most obscure conditions that could lead to a system compromise, and allow anyone to alert the world to new threats. However, some lists are regularly very busy. Meanwhile, the www.sans.org website provides education and support to implement best security practices.



The CERT Internet Storm Center is a focal point for information and analysis about growing and receding threats posed to the internet

The Internet Storm Center is a goldmine of information about the current threats on the internet, and even the sources of those threats, along with information about vulnerabilities and saying safe.

It's clear from the need for these groups, and the fact that international law is being tightened, that computer crime is not going away. Underlying much illegal activity is an evolving list of susceptible software that makes it essential to protect against threats not covered by your firewall or IDS.

NEXT MONTH

LINUX ALTERNATIVES

Linux is powerful and flexible, but is it always the best operating system for the job? We look at other UNIX-based options